TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2006

END-YEAR

COMP 310

Concurrent Programming

Time Allowed: 3 Hours

Instructions: Answer all **six** questions.

Each question is worth 30 marks.

The exam is worth 180 marks in total.

Paper foreign language dictionaries are permitted.

Non-programmable calculators without full alphabetic keys are permitted.

Electronic dictionaries and programmable calculators are not permitted.

Question 1. General Concepts of Concurrency [30 marks]

(a) [10 marks] In studying concurrency, we usually assume that an execution of a concurrent program is an arbitrary interleaving of atomic statements from individual processes. Explain why this abstraction is appropriate in each of the following contexts:

(i) Multitasking systems

(ii) Multiprocessor systems

(iii) Distributed systems

(b) [10 marks] Use a state diagram to determine the possible outcomes of the program $x = x + 1 \;//\; x = x + 2$ under each of the following assumptions about the machine on which the program is executed:

(i) The machine has an atomic Add To Memory instruction.

(ii) Memory can only be accessed using atomic load and store instructions.

(c) [10 marks] Briefly discuss the advantages and disadvantages of each of the following approaches to determining the correctness of concurrent programs:

(i) Testing

(ii) Model checking

(iii) Formal verification

Your answer should address the application of these approaches to both safety and liveness properties.

Question 2. Critical Sections [30 marks]

(a) [6 marks] A solution to the critical section problem is required to satisfy the following correctness properties:

(i) Mutual exclusion

(ii) Deadlock freedom

(iii) Freedom from individual starvation

Explain what is meant by each of these properties and why they are required.

(b) [16 marks] Consider the following algorithm, which is a proposed solution to the critical section problem:

Critical section			
boolean wantp ← false, wantq ← false			
p		**q**	
	loop forever		loop forever
p1:	non-critical section	q1:	non-critical section
p2:	wantp ← true	q2:	wantq ← true
p3:	while wantq	q3:	while wantp
p4:	wantp ← false	q4:	wantq ← false
p5:	wantp ← true	q5:	wantq ← true
p6:	critical section	q6:	critical section
p7:	wantp ← false	q7:	wantq ← false

For each of the correctness properties listed in part **(a)** above, explain why this algorithm does or does not satisfy. If the property is not satisfied, explain how the algorithm can be modified so that it does satisfy the property.

(c) [8 marks] Explain how the algorithm given in part **(b)** above can be implemented using semaphores.

Question 3. Monitors and Queues [30 marks]

(a) [10 marks] Explain briefly what is meant by a *monitor*, and show how a concurrent queue can be implemented using a monitor.

(b) [6 marks] Explain briefly how you would implement this monitor in Java.

(c) [4 marks] Why does the simple use of a monitor to implement a concurrent queue restrict the possibilities for parallel execution?

(d) [10 marks] Describe an alternative implementation of a concurrent queue, which provides greater opportunities for parallel execution.

Question 4. Process-based Synchronization. [30 marks]

(a) [5 marks] Identify the main alternatives in the design of constructs for synchronization by communication.

Communications may be synchronous or asynchronous.
In synchronous communication the sender blocks until the receiver receives whereas in asynchronous the sender does not block.
Addressing may be symmetric (both client and server have addresses), assymmetric (only server has an address) or not required.
Dataflow may be one way or two. Synchronous may be either and asynchronous is by definition one-way.

(b) Consider synchronous channels.

(i) [4 marks] Define the semantics of the notation used below:

```
ch ⇐ value
ch ⇒ variable
```

ch is the name of the channel. If destination is attempting to send a message using the channel and source is attempting to receive from the channel then the values is assigned to the variable. Both statements block until both are available. There is no buffering.

(ii) [4 marks] Explain the operation of the following pseudo-code (note that we assume two channels `ch1` and `ch2` have already been defined as well as a variable `x`):

```
either
  ch1 ⇒ x
or
  ch2 ⇒ x
```

Only one alternative can succeed.
If communication can take both on multiple channels then one is chosen non-deterministically and succeeds (either ch1 or ch2).
Otherwise it is the alternative branch with the communication statement that could succeed.
If neither can succeed the process blocks.

(iii) [12 marks] Using the notation discussed above, write an algorithm for pipeline sort. There are n processes and n numbers fed into the input channel of the first process. The value 255 is used to mark the end of the number stream and to indicate that the algorithm should terminate. When the algorithm terminates, the numbers are stored in *descending* order in the processes. For example:

(c) [5 marks] Evaluate the claim that workers in a heartbeat algorithm implemented using synchronous channels cannot get more than one iteration ahead of other workers.

Immediate neighbours cannot because they are dependent upon the values from neighbours for their calculation. However, neighbours further apart may be dependent upon different inputs that follow different paths and so can get further ahead. The key issue is the flow of messages because each worker is an independent process and if it has the data it needs, it can perform its calculation.

Question 5. Data-based Synchronization. [30 marks]

This question assumes that all the Linda `readnote` and `removenote` primitives are blocking.

(a) The following is an outline of an algorithm implementing a client server program. Note that there may be more than one client but only one server.

Client Server	
client	**server**
constant integer me ← ...	integer client
serviceType service	serviceType s
dataType result, parm	dataType r,p
p1: service ← // Service requested	q1: removenote('S', client, s, p)
p2: postnote('S', me, service, parm)	q2: r ← do(s, p)
p3: removenote('R', me, result)	q3: postnote('R', client, r)

(i) [6 marks] Use this example to explain the semantics of `postnote` and `removenote`.

Postnote: creates a note from the vlaues of the parameters and posts it into the space. If there are processes blocked waiting for a note machines, an arbitrary one of them is unblocked. Removenote: the parameters are variables. Removes a note that matches the parameter signature from the space and assigns it values to the parameters. If no matching note exists, it blocks. (6 marks)

(ii) [4 marks] Outline and explain changes to the code above that are necessary to allow it to *select* a request for one particular service.

Current the server accepts any request. Need to use formal parameter or post unwanted requests back into the space. Formal parameters just requires removenote('S', client, s=, p). This will only match notes containing the current value of the variable. (4 marks)

(b) [5 marks] Describe how you would implement a general semaphore in Linda. What Linda primitive plays the role of *wait*, and what Linda primitive plays the role of *signal*?

do K times: postnote('s'). removenote is equivalent to wait because will block and postnote to signal because will unblock.

(c) [10 marks] Explain how you would implement matrix multiplication in Linda using the master-worker paradigm. You are not required to give a full implementation including all the detail in the text or covered in lectures. Provide sufficient detail to indicate how the process interaction is structured, and the task performed by each process. Describe any advantages gained by structuring a computation this way.

Master-worker – master puts the tasks into the space. Workers remove them and process them, results are posted back into the space. Each worker could process a row and column indepdently of other workers. The master could populate the space and block until results start arriving. The master could then print them out. Advantages: relative speeds of workers not important, can add more workers at any time. can remove workers as long as they have not removed a task.

(d) [5 marks] Contrast the style of coupling between processes imposed by synchronous message passing with the style imposed by the Linda model.

Synchronous are tight coupled in time and space. Sender and receiver must exist at same time and indentity of channel must be available to both. With Linda, the processes are decoupled. Sender and receiver can be present at different times, do not need to agree upon identities.

Question 6. Mutual Exclusion [30 marks]

(a) [5 marks] Consider the full permission-based Ricart-Agrawala algorithm (RA algorithm). Describe how a node is prevented from entering its critical section by another node that holds a lower ticket number.

The node holding the lower ticket number does not send a reply message. Because the requesting node requires replies from all other nodes it is blocked until this node (and the others) reply.

(b) [9 marks] Explain how the full permission-based RA algorithm prevents the following scenarios:

(i) Two nodes choose the same ticket number, leading to deadlock of the entire algorithm.

Make the algorithm asymmetric by using NodeID (each node is assumed to have a unique ID) as a tie break. This ensures that cannot be tied on the same ticket number.

(ii) A node keeps choosing a ticket number smaller than all the other ticket numbers chosen by other nodes, leading to violation of mutual exclusion.

Force nodes to choose monotonically increasing timestamps. Each node remembers the highest node ID seen and chooses an ID that is larger

(iii) One node never requests entry to its critical section, leading to deadlock of the entire algorithm because other nodes are waiting for its permission to enter their critical section.

If a node does not want entry to the critical section, it immediately answers yes otherwise ticket numbers are compared to determine who should enter first. This prevents a node who would be allowed to enter first (because either it never asked or it has been a long time since it made a request) but does not want entry from holding up other nodes who do want entry.

(c) [6 marks] Consider a node k that is one of N nodes in a distributed system. The node has just exited its critical section and wishes to re-enter it. No other node has requested entry to its critical section since node k exited its critical section.

Calculate, for both the permission-based RA algorithm and token-based version, the number of request and reply messages sent before node k is allowed to re-enter its critical section. Make sure that you explain the basis of your calculations.

Permission-based: N-1 request messages and N-1 reply messages because it requires permission from every other node before it can enter (3 marks). Token-based: no-one else has requested entry so there is no need to send any messages because this node is the only one who may be in the critical section due to possession of the token and no other node has requested entry (3 marks).

(d) [5 marks] Explain why a node can receive request messages out of order when executing the token-based RA algorithm.

Because in the token-based RA algorithm, the requesting process can proceed once it receives the token from any one of the nodes and further requests generated. This means that other nodes who do not hold the token may receive multiple requests and these could be reorded by the underlying communications system.

(e) [5 marks] Explain why the token-passing RA algorithm is vulnerable to starvation, and propose a change to the algorithm to ensure that starvation cannot occur.

The algorithm chooses an arbitrary node from those waiting to send the token to. This could mean that one node never gets chosen. One fix would be to use a queue. Whenever a request is made, new requesting nodes are added to the end of the queue and removed from the front when granted. An alternative approach would be to keep track of the number of times granted and ensure an even number is always used.

****